A Comparison of Acceptance Criteria for the Daily Car-Pooling Problem

Jerry Swan[1], John Drake[1], Ender Özcan[1], James Goulding[2], John Woodward[1]

[1]Automated Scheduling, Optimisation and Planning (ASAP) Research Group,
[2]Horizon Digital Economy Research Institute,
School of Computer Science, University of Nottingham,
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK.
{jps,jqd,exo,jog,jrw}@cs.nott.ac.uk

Abstract. Previous work on the Daily Car-Pooling problem includes an algorithm that consists of greedy assignment alternating with random perturbation. In this study, we examine the effect of varying the move acceptance policy, specifically Late-acceptance criteria with and without reheating. Late acceptance-based move acceptance criteria were chosen because there is strong empirical evidence in the literature indicating their superiority. Late-acceptance compares the objective values of the current solution with one which was obtained at a fixed number of steps prior to the current step during the search process in order to make an acceptance decision. We observe that the Late-acceptance criteria also achieve superior results in over 75% of cases for the Daily Car-Pooling problem, the majority of these results being statistically significant.

1 Introduction

There is increasing economic and environmental interest in minimizing the consumption and emission of petrochemicals arising from the use of personal vehicular transport. The task of assigning passengers to drivers in order to increase vehicle occupancy while minimizing the additional journey length incurred is known as the Daily Car-Pooling problem (DCPP). The DCPP can be considered to be a generalization of the Dial-A-Ride Problem (DARP) in which the vehicles are heterogeneous [3]. As is the case with all variants of the Vehicle Routing Problem (VRP), it is known to be NP-hard [13]. We can consider the DCPP to be a VRP with the additional constraints of Pickup and Delivery (VRPPD) and Time Windows (VRPTW), the latter being an example of *quality of service* criteria in which we seek to minimize the inconvenience suffered by all participants.

The specific problem addressed in this paper is the *'To Work' DCPP*, in which a pool of users (employees) participate in vehicle sharing for travel to a central destination (the workplace). Some members of the pool are designated to be drivers (*servers*), the remainder are passengers (*clients*). All employees have a time window in which their journey must take place. Servers stipulate

a maximum journey time and their vehicle has an associated capacity. Clients not assigned to any server have an associated penalty. The objective is to assign clients to servers so as to minimize the sum of the total distance travelled and the penalties incurred for unserviced clients. In the instances considered here, all quantities are integers. In [6], Cordeau and Laporte give an extensive overview of the DARP and observe that two decades of research has made it routinely possible to schedule hundreds of employees. Cordeau and Laporte also observe that approaches can be differentiated according as they perform clustering and routing as distinct, sequential phases or whether they interleave these activities. In Baldacci et al. [1], the DARP is solved by Lagrangean relaxation. If the location of all participants are known *a priori*, the DCCP is said to be *static*. We restrict ourselves to the static case. In [3], Calvo et al. give an algorithm for the DCPP (an adaptation of the capacitated p-median algorithm in [9]) that consists of greedy assignment alternating with random perturbation. The greedy assignment phase proceeds by seeking to minimize a marginal quantity termed *regret* — an estimate of the total extra mileage that would be incurred over all journeys for a passenger.

Maniezzo et al. [8] describe a solution to the Long-Term Car Pooling Problem (a variant in which the role of driver alternates between pool members) that employs Ant-Colony Optimization (ACO). ACO is an example of a *metaheuristic* technique that seeks to perturb candidate solutions beyond local optima. It is clearly also possible to apply other metaheuristic techniques such as Simulated Annealing, Evolutionary Strategies, Genetic Algorithms and Tabu Search.

Recent research [10] indicates that the choice of acceptance criterion is one of the more significant metaheuristic mechanisms. In this article, we examine the effect of varying the acceptance criterion. In particular, we investigate the use of *Late-acceptance Hillclimbing*. The Late-acceptance Hillclimbing (LA) metaheuristic [2] is a simple yet suprisingly effective strategy — a new solution is accepted if it is no worse than the k-th most recent incumbent solution. The stated advantages of the LA are that it is reliant on only the single parameter k; it is not sensitive to initialisation and has been shown to be superior to (or at least competitive with) best-known results in a number of domains (e.g. [2],[14],[11]).

2 Experimental Framework

For our experiments, we made use of the HYPERION framework [12], implemented in the Java programming language. HYPERION provides a combinatorial optimization framework parameterized by concepts of STATE, LOCALITY (i.e. neighbourhood) and OBJECTIVE FUNCTION. We used datasets A and B as described in [1]. Class A is an adaptation of the datasets of [4] [5] and [7] and consists of 12 problems with the number of employees ranging from 50 to 225. Class B is adapted from real-world data and consists of 23 problems with the number of employees ranging from 100 to 250. The authors state that both classes of problems are intended to simulate real-world applications.

```
JourneyMatches hdcpp( List<Server> servers, List<Client> clients )
{
  // phase 1
  JourneyMatches matches = hdcppPhase1 ( servers, clients) ;
  // phase 2
  double bestValue = objectiveFn ( matches ) ;
  JourneyMatches bestMatching = matches ;

  initializeAcceptanceCriterion( bestValue ) ;

  long numUnimprovingIter = 0 ;
  for ( long iter = 0 ; ; ++iter )
  {
      JourneyMatches newMatches = hdcppPhase2InnerLoop( matches ) ;
      double currentValue = objectiveFn. valueOf( matches ) ;
      double newValue = objectiveFn. valueOf( newMatches ) ;
      boolean accept = acceptanceCriterion( currentValue, newValue, iter ) ;
      if ( accept )
      {
          matches = newMatches ;
          if ( currentValue < bestValue )
          {
              bestValue = currentValue ;
              bestMatching = matches ;
          }
      }
      if ( currentValue >= bestValue )
          ++numUnimprovingIter ;
      if ( terminationCondition(   matches, currentValue, newMatches,
                                   newValue, iter, numUnimprovingIter )  )
          break ;
      // update state of acceptanceCriterion
      // e.g. reheat etc. as appropriate
      acceptanceCriterion.update( ) ;
  }
}
```

Fig. 1. Algorithm DCPP

We configured HYPERION with a STATE given by the pair (J, U) where J is the set of *Journies* (i.e. the set of assignments of clients to servers) and U is the set of unmatched clients. The LOCALITY is identical to that employed by Calvo et al., viz. the set of all states reachable from the present one via the unmatching of a single passenger. The objective function to minimize is then given by the total path cost plus the sum of the penalties incurred by unmatched passengers. We configured the framework with acceptance policies *Improving or Equal* (IE), *Late-acceptance* (LA) and *Late-acceptance with reheating* (LR). Figure 1 describes the top-level of the experimental framework: the essential difference from the pseudocode given in [3] is the addition of extension points to provide for the initialization, evaluation and internal-state update of the variant acceptance criteria. These extension points operate as follows: the IE policy requires no additional initialization or state-update, and accepts new values that are greater than or equal to the current value. The operation of the late-acceptance

policies is derived from [2]: initialization of the late-acceptance policies involves the creation of a history list of fixed length, the values of which are given by o_1, the objective value obtained from phase 1 of the matching. Both late-acceptance policies accept a new value if it is better than or equal to either the current value or the k-th most recent value, where k is given by the iteration count modulo the length of the history list. An accepted solution replaces the solution to which it is being compared in the history list. Whenever the number of non-accepting iterations exceeds the threshold parameter MAX_IDLE_ITER, the internal-state update for LARH achieves reheating by adding an offset to all entries in the history list, which in our experiments was fixed at 0.1 * o_1.

3 Results

Table 1 gives $(\bar{x}, \sigma^2)$ of the objective function value obtained with the IE, LA and LR acceptance criteria for 30 runs of the A and B datasets. The 'label' and 'size' columns give the instance name and number of employees, respectively. The termination criterion was 10,000 un-improving moves (decided experimentally). LA and LR have a history list length of 100, MAX_IDLE_ITER for LR was set to 200. Table 1 also compares the late-acceptance criteria for statistical significance (t-test with $p = 0.05$) against the IE criterion. For acceptance criteria A_1 and A_2, $A_1 \geq A_2$ indicates A_1 outperforms A_2 on average whilst $\gg$ indicates that this difference is statistically significant (conversely $\leq$ and $\ll$). It can be seen from these tables that there is relatively low variance in solution quality: this may be due to the constructive phase resulting in a basin of attraction in the solution-space.

LA outperforms IE in 77.1% of instances, specifically 7 out of 12 A instances and 20 out of 23 B instances. 63% of this performance difference is statistically significant (3 out of 7 A instances and 14 out of 20 B instances). LR outperforms IE in 88.6% of instances, specifically 9 out of 12 A instances and 22 out of 23 B instances. 54.8% of this performance difference is statistically significant (5 out of 9 A instances and 12 out of 22 B instances). LR is therefore performs particularly well on dataset B. It is interesting that for instances A05 and A10 the IE strategy outperforms both late-acceptance criteria. Instance B03 is also interesting as LA performs worse than IE, but LR performs significantly better than it.

4 Conclusion

We have applied two variants of late-acceptance hillclimbing to a greedy algorithm for the Daily-Car Pooling Problem and compared them with naïve hill-climbing. Both late-acceptance strategies are superior to the naïve approach in most cases and this is often statistically significant for larger instances.

The superior performance of the reheating variant of late-acceptance can be explained in part by the 'fitness-cycling' effect of reheating, which generally means that there are a larger number of iterations before the 'number-of-

Table 1. (Mean, standard deviation) and t-test comparison against IE criterion (vs) of objective function values of 30 runs of the datasets A and B by acceptance criterion

label	size	LA	vs	LR	vs	IE
A01	50	(1202,77)	$\geq$	(1190,54)	$\gg$	(1224,79)
A02	75	(1638,87)	$\leq$	(1619,62)	$\geq$	(1619,90)
A03	100	(1459,79)	$\geq$	(1461,104)	$\geq$	(1502,85)
A04	120	(2381,22)	$\gg$	(2387,34)	$\gg$	(2438,38)
A05	120	(2318,145)	$\ll$	(2253,123)	$\ll$	(2120,85)
A06	134	(2472,101)	$\leq$	(2453,123)	$\geq$	(2456,84)
A07	150	(2372,142)	$\geq$	(2353,144)	$\gg$	(2446,209)
A08	170	(2976,76)	$\leq$	(2972,61)	$\leq$	(2955,45)
A09	170	(2777,48)	$\geq$	(2770,57)	$\geq$	(2857,74)
A10	195	(3397,101)	$\ll$	(3357,95)	$\ll$	(3288,36)
A11	199	(2060,61)	$\gg$	(2040,63)	$\gg$	(2117,96)
A12	225	(2345,52)	$\gg$	(2335,43)	$\gg$	(2435,102)
B01	100	(1704,76)	$\gg$	(1718,80)	$\geq$	(1743,82)
B02	100	(1531,23)	$\geq$	(1531,27)	$\geq$	(1542,33)
B03	100	(1697,129)	$\leq$	(1615,109)	$\gg$	(1688,124)
B04	100	(2255,23)	$\leq$	(2255,32)	$\leq$	(2255,30)
B05	100	(1910,105)	$\gg$	(1932,107)	$\gg$	(2021,118)
B06	100	(1477,61)	$\geq$	(1464,84)	$\geq$	(1491,78)
B07	100	(1343,36)	$\gg$	(1360,33)	$\gg$	(1386,58)
B08	150	(2047,47)	$\gg$	(2036,39)	$\gg$	(2081,69)
B09	150	(1980,36)	$\gg$	(1987,37)	$\gg$	(2018,47)
B10	150	(2768,88)	$\geq$	(2787,93)	$\geq$	(2808,110)
B11	150	(2217,112)	$\gg$	(2254,112)	$\geq$	(2283,144)
B12	150	(1866,82)	$\geq$	(1855,90)	$\geq$	(1883,96)
B13	200	(2706,59)	$\gg$	(2691,99)	$\gg$	(2793,107)
B14	200	(2689,104)	$\geq$	(2703,94)	$\geq$	(2722,131)
B15	200	(3467,66)	$\gg$	(3463,69)	$\gg$	(3524,81)
B16	200	(3690,100)	$\leq$	(3686,111)	$\geq$	(3686,112)
B17	200	(4111,126)	$\gg$	(4135,128)	$\gg$	(4249,115)
B18	200	(2716,111)	$\gg$	(2750,77)	$\geq$	(2786,91)
B19	250	(3542,82)	$\gg$	(3566,79)	$\geq$	(3592,105)
B20	250	(3680,108)	$\geq$	(3633,103)	$\gg$	(3696,126)
B21	250	(3869,47)	$\gg$	(3872,58)	$\gg$	(4024,112)
B22	250	(3732,95)	$\gg$	(3707,91)	$\gg$	(3805,113)
B23	250	(3436,121)	$\gg$	(3404,119)	$\gg$	(3552,117)

unimproving-moves' termination criterion is met. Future work will attempt to gain further insight into the outlying instances (A05,A10,B03) and attempt to further distinguish between the two late-acceptance strategies: in particular, we anticipate that LR would perform significantly better than LA in almost all cases if the number of experiments were increased.

References

1. Roberto Baldacci, Vittorio Maniezzo, and Aristide Mingozzi. An exact method for the car pooling problem based on Lagrangean column generation. *Operations Research*, 52(3):422–439, 2004.
2. Edmund K. Burke and Yuri Bykov. A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems. In *PATAT '08*, 2008.
3. Roberto Wolfler Calvo, Fabio de Luigi, Palle Haastrup, and Vittorio Maniezzo. A distributed geographic information system for the daily car pooling problem. *Comput. Oper. Res.*, 31(13):2263–2278, 2004.
4. N Christofides and S Eilon. An algorithm for the vehicle dispatching problem. *Operational Research Quarterly*, 309(20), 1969.
5. N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In *Combinatorial Optimization*. Wiley, 1979.
6. Jean-Franois Cordeau and Gilbert Laporte. The Dial-a-Ride Problem (darp): Variants, modeling issues and algorithms. *4OR*, 1:89–101, 2003.
7. M. L. Fisher. Optimal solution of vehicle routing problems using minimum K-trees. *Operations Research*, 42(4):626–642, 1994.
8. V. Maniezzo, A. Carbonaro, and H. Hildmann. *New Optimization Techniques in Engineering*, chapter An ANTS heuristic for the Long-Term Car-Pooling Problem. Springer, 2002.
9. John M. Mulvey and Michael P. Beck. Solving capacitated clustering problems. *European Journal of Operational Research*, 18(3):339 – 348, 1984.
10. Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz. A comprehensive analysis of hyper-heuristics. *Intell. Data Anal.*, 12(1):3–23, 2008.
11. Ender Özcan, Yuri Bykov, Murat Birben, and Edmund K. Burke. Examination timetabling using late acceptance hyper-heuristics. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 997–1004, Piscataway, NJ, USA, 2009. IEEE Press.
12. Jerry Swan, Ender Özcan, and Graham Kendall. Hyperion - a recursive hyper-heuristic framework. In Carlos A. Coello Coello, editor, *Learning and Intelligent OptimizatioN, 5th International Conference, LION 5*, LNCS, 2011.
13. P. Toth and D. Vigo. An overview of vehicle routing problems. In *The vehicle routing problem*, pages 1–26. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.
14. Jannes Verstichel and Greet Vanden Berghe. A late acceptance algorithm for the lock scheduling problem. In Stefan Voss, Julia Pahl, and Silvia Schwarze, editors, *Logistik Management*, pages 457–478. Physica-Verlag HD, 2009.